VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vytváření databáze poruch v elektrických sítích

Building Database of Failures in Power Networks

2020 Marián Grobář

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Marián Grobář**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vytváření databáze poruch v elektrických sítích
Building Database of Failures in Power Networks

Jazyk vypracování: čeština

Zásady pro vypracování:

Na Katedře informatiky je vyvíjen informační systém pro analýzu dat poruch v elektrických sítích, který je využíván distributory elektrické energie v České i Slovenské republice. Jednotlivý distributoři poskytují svá data v různých formátech, z těchto dat je pak vytvářena jednotná databáze nad kterou jsou prováděny spolehlivostní výpočty. Vytváření jednotné databáze aktuálně probíhá v samostatné desktopové aplikaci, spolehlivostí výpočty jsou integrovány v aplikaci webové. Cílem práce je integrovat vytváření databáze do webové aplikace.

1. Nastudujte stávající webovou i desktopovou aplikaci a databázi poruch.
2. Navrhněte a naimplementujte integraci vytváření databáze do webové aplikace.
3. Výslednou aplikaci vyhodnoťte a porovnejte.

Seznam doporučené odborné literatury:

[1] R. Goňo, M. Krátký, S. Rusek: Analysis of distribution network failure databases.
Przeglad elektrotechniczny 86 (8), 168-171, 2010.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Michal Krátký, Ph.D.**

Datum zadání: 01.09.2019
Datum odevzdání: 30.04.2020

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 15. května 2020 ..... Jakub

Rád bych na tomto místě poděkoval vedoucímu práce doc. Ing. Michalovi Krátkému, Ph.D. za odborné vedení a pomoc při realizaci této práce.

Abstrakt

Cílem této bakalářské práce je implementovat vytváření jednotné databáze informačního systému, který je určen pro analýzu dat poruch prvků elektrických sítí. Data potřebná pro vytvoření databáze jsou dodána od jednotlivých distributorů elektrické energie. Dříve bylo možné vytváření databáze jen v desktopové aplikaci. Vytváření databáze ve webové aplikaci zjednoduší proces vytváření a aktualizace databáze. Implementace bude zahrnovat také návrh uživatelského rozhraní, které se bude pro tyto účely využívat. Uživatel bude mít k dispozici přehled s relevantními informacemi pro aktualizaci databáze.

Klíčová slova: SQL, Webová aplikace, SQL Server, Databázové systémy, HTML, SQLite

Abstract

The aim of this bachelor thesis is implementation of creating unified database for information system, whose main puprose is data analysis of equipment failures on the electrical network. Data needed for database creation are delivered by individual electricity distributors. Before, the creation of database was possible only in the desktop application. Creation of database in the web application will ease the process of creation of databse and its updating. Implementation will contain user interface design, which will be used for the purpose of the database update. User will have all the relevant information about the update of the database available.

Keywords: SQL, Web application, SQL Server, Databse system, HTML, SQLite

Obsah

Seznam použitých zkratek a symbolů

SQL	–	Structured Query Language
SSMS	–	SQL Server Management Studio
HTML	–	Hypertext Markup Language
HTTP	–	Hypertext Transfer Protocol
SŘBD	–	Systém řízení báze dat
DML	–	Data Manipulation Language
DDL	–	Data Definition Language
PHP	–	Hypertext Preprocessor
ASP	–	Active Server Pages
SPA	–	Single-page application
AJAX	–	Asynchronous JavaScript and XML
API	–	Application Programming Interface
JS	–	JavaScript
CPU	–	Central processing unit
ORM	–	Object-Relational-Mapper
T-SQL	–	Transact-SQL
CSS	–	Cascading Style Sheets
XML	–	Extensible Markup Language
SVG	–	Scalable Vector Graphics
DOM	–	Document Object Model
JSON	–	JavaScript Object Notation
CLR	–	Common Language Runtime
CIL	–	Common Intermediate Language
JIT	–	Just in Time
IBM	–	International Business Machines

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Katedra informatiky spravuje informační systém pro výpočet spolehlivostních parametrů prvků elektrických sítí nad daty poskytovanými od různých distributorů elektrické energie. Aby byly výpočty možné, tato data se musí nejprve zpracovat a uložit do jednotné databáze. Jednotná databáze byla vytvořena z důvodu odlišných schémat, které distributoři používají. Momentálně existují dvě verze systému, webová aplikace a desktopová aplikace. V tuto chvíli je vytvoření databáze možné jen v desktopové aplikaci. Ta využívá databázový systém RadegastDB, který je vyvíjen výzkumnou skupinou Katedry informatiky, zatímco webová aplikace slouží k výpočtu spolehlivostních parametrů prvků. Hlavním cílem bakalářské práce je integrovat vytváření jednotné databáze do webové aplikace. Na rozdíl od desktopové verze, webová aplikace nyní používá pro uložení dat databázový systém SQL Server [4] od společnosti Microsoft [42], který nahradil předešlý databázový systém RadegastDB. Tato úprava usnadní administrátorům aktualizaci stávající databáze. V současné době pro aktualizaci databáze by se nejprve pomocí desktopové aplikace muselo provést zpracování záznamů a poté se tyto záznamy musí importovat do databáze webové aplikace. Nejprve bylo nutné analyzovat stávající informační systém a jeho datovou strukturu, poté navrhnout nové relace, které budou využívány pro indexaci dat. Následně bylo potřebné nastudovat algoritmus pro vytváření databáze a provést nutné úpravy. Další krokem bylo vytvoření uživatelského rozhraní sloužícího pro indexaci a správu souborů potřebných k vytvoření databáze.

Obsah bakalářské práce je rozčleněn do 5 kapitol.

Kapitola 2 obsahuje základní informace k databázovým systémům a jejich využití, také se čtenář seznámí s pojmem webová aplikace, architekturou webových aplikací a porovnání s desktopovou aplikací.

Kapitola 3 nastíní čtenáři stávající systém pro analýzu dat, jeho datovou strukturu, průběh zpracování dat a funkce jednotlivých formulářů.

Kapitola 4 popisuje analýzu řešeného problému a implementaci úprav, které bylo zapotřebí udělat. Poslední kapitola obsahuje zhodnocení provedených úprav a porovnání se stávající aplikací.

2 Databázové systémy a webové aplikace

2.1 Databázové systémy

Databáze je organizovaný soubor strukturovaných dat, které se obvykle ukládají v elektronické podobě v počítačovém systému [44]. Databáze je obvykle řízena systémem který umožňuje databázi definovat, konstruovat a manipulovat s ní. Tento systém se nazývá systém řízení báze dat [40] (SŘBD, angl. DBMS). Data a SŘBD se označují jako databázový systém. Databázové systémy můžeme podle architektury dělit na embedded a klient-server.

Architektura klient-server znamená, že databázový systém je tvořen dvěma částmi - serverovou a klientskou. Klient komunikuje s databází přes počítačovou síť. Serverová část spravuje všechna data a vykonává operace nad nimi. Druhou architekturou jsou Embedded databázové systémy.

Embedded databázové systémy běží v adresářovém prostoru aplikace, která je využívá. Embedded databázový systém je knihovna, kterou si přilinkujete k aplikaci a pomocí této knihovny přistupujete přímo k souborům s daty. Z tohoto přístupu k datům je patrné, že tyto databáze jsou hlavně určeny pro jedno uživatelské aplikace. Embedded databázové systémy vyžadují takřka nulovou údržbu, na rozdíl od serverových, protože o většinu se stará samotná aplikace.

SQLite [23] je nejvíce používaný databázový systém na světě [36]. Najdeme jí například v každém Android zařízení [38], v každé instalaci Windows 10 [21] nebo také ve webovém prohlížeči Chrome [20]. V současné době existuje přes 1 trilion SQLite instancí, které jsou aktivní. SQLite poskytuje pouze nezbytné základy z jazyka SQL [34], jako jsou například příkazy pro manipulaci s daty nebo příkazy pro dotazování, kromě příkazů `RIGHT OUTER JOIN` a `FULL OUTER JOIN`. Čtení a zápis informací probíhá přímo na disk. Výkon SQLite je však závislý na velikosti přidělené operační paměti [23]. Celá databáze je reprezentována jedním souborem, který je přenositelný napříč různými platformami.

2.1.1 Relační databázové systémy

V roce 1970 vydal Edgar F.Codd práci s názvem "A Relational Model of Data for Large Shared Data Banks"[43], popisující teorii relačních databází. Na základě této práce poté v vznikly první relační databázové systémy (RDBMS), System R od společnosti IBM [46] a Ingres, který byl vytvořen na Kalifornské univerzitě [45]. V 80. letech se RDBMS dostali na vrchol a začala upadávat popularita hierarchických databázových systémů. V dnešní době jsou stále relační databázové systémy nejvíce využívaným typem databázových systémů.

Pod pojmem relační databáze si můžeme představit tabulky (relace) s řádky a sloupci, které mají mezi sebou vztahy. Jsou založeny na relačním datovém modelu, intuitivním, přímočarém způsobu vyjádření dat v tabulkách. Tabulky slouží k uložení informací o objektech která jsou v databázi. Každý sloupec v tabulce obsahuje určitý typ dat a buňka obsahuje hodnotu atributu. Záznamy v tabulce reprezentují seznam hodnot, které se sebou souvisí a patří k jednomu objektu

nebo entitě. Každý řádek v tabulce může být označen jedinečným identifikátorem nazývaným primární klíč a řádky mezi více tabulkami mohou být propojeny pomocí cizích klíčů.

S relačním datovým modelem je také úzce spojen jazyk SQL [34]. SQL neboli strukturovaný dotazovací jazyk je dotazovací jazyk, který se používá pro komunikaci s relační databází a je založen na relační algebře. SQL se stalo standardem v roce 1986. Standard byl vydán americkým institutem ANSI [37] pod název SQL-86. V dnešní době je SQL standard rozdělen na 10 částí. V nejnovějším standardu byly přidány vícerozměrná pole. Většina SŘBD, které používají SQL, mají také vlastní modifikace tohoto jazyka, které fungují jen na jejich systémech. SQL je používán k modifikaci dat, dotazování nad daty, analýze aplikací a transakčnímu zpracování.

V transakčním zpracování dat se používá pojem transakce. Tento pojem označuje souhrn příkazů, které jsou brány jako jeden celek a také by podle toho měly být vykonány. Transakce začíná BEGIN TRANSACTION a končí operacemi COMMIT (úspěšné zpracování transakce) nebo ROLLBACK (neúspěšné zpracování transakce) [40]. Jedna transakce se může skládat z několika příkazů. Pokud nějaký z příkazů selže, změny provedené předchozími příkazy jsou neplatné (provede se ROLLBACK). Z tohoto je tedy patrné, že transakce může skončit jen dvěma způsoby. Buď se transakce vykoná správně, v tom případě se potvrdí všechny změny (COMMIT), nebo nastane chyba při vykonávání a provedené změny se zruší (ROLLBACK). Transakce slouží jako ochrana před chybami, které se mohou vyskytovat v systému (nedostatek místa na disku, výpadek systému). Každá transakce musí splňovat čtyři základní vlastnosti - ACID [40].

1. Atomicita (atomicity) - aby transakce byla úspěšná musejí být provedeny všechny příkazy. K zrušení transakce může dojít kdykoliv před jejím potvrzením. V takovém případě všechny uskutečněné změny budou vráceny do původního stavu.

2. Konzistence (consistency) - příkazy v rámci transakce převádějí konzistentní stav databáze do jiného konzistentního stavu.

3. Izolace (isolation) - změny prováděné v transakci nejsou viditelné jiné transakci dokud nedojde k potvrzení.

4. Trvalost (durability) - v momentě kdy je transakce potvrzena, všechny provedené změny se stávají trvalými.

2.1.2 SŘBD

SŘBD je program umožňující ukládat data a dotazovat se na uživatelské data. Skládá se ze skupiny programů které manipulují s databází. SŘBD obdrží požadavek na data a vyšle operačnímu systému instrukci aby tato data poskytl. Nejznámější databázové systémy jsou MySQL [1], Microsoft Access [3], Microsoft SQL Server [4], FileMaker Pro [5], Oracle Database [27] a dBASE [6].

Funkce SŘBD:

1. Dovoluje uživatelům vytvářet nové databáze a specifikovat jejich schéma (logická struktura dat), za použití speciálního jazyka určeného pro definici dat (DDL - data definiton language).

2. Dává uživatelům schopnost dotazovat se na data a upravovat data, s použitím jazyka který je určený na dotazování do databáze (query language) a jazyka který slouží k manipulaci s daty (DML - data manipulation language).

3. Podporuje ukládání velkého objemu dat po dlouho dobu, což umožňuje efektivní přístup k datům pro dotazy a úpravy databáze.

4. Umožňuje obnovu databáze v případě selhání, chyby nebo úmyslnému zneužití.

5. Řídí přístup k datům od mnoha uživatelů najednou, aniž by umožňoval neočekávané interakce mezi uživateli (izolace) a operace nad daty jsou provedeny všechny nebo žádné (atomičnost).

2.1.3 Využití databázových systémů

V dnešní době jsou databázové systémy neodmyslitelnou součástí informačních systému. Databázové systémy umožňují sdílet data napříč celým světem. Sběr a zpracování dat je dnes takřka v každém pracovním odvětví. Banky, nemocnice, statní instituce, obchody a mnoho dalších podniků dnes využívá informační systémy za účelem shromáždění a poté následné prezentaci dat. Data uložená v databázích umožňují například firmám optimalizovat jejich firemní procesy nebo zlepšit řízení podniku. Skoro každý člověk na světě dnes používá databázový systém. Říkáte si jak je to možné? Můžou za to mobilní telefony. Každý mobilní telefon používá nějakým způsobem databázi. Konkrétněji SQLite [23] tento typ databázového systému se používá ve všech Android [38] a iOS [39] zařízeních. Využití databázových systémů rok od roku roste a také se stále vyvíjí nové typy databázových systémů. V dnešní době jsou velice zajímavou technologií autonomně řízené databáze [35]. Tyto databáze používají k provádění základních administrátorských úloh strojové učení a odpadá tak množství operací, které musí administrátor vykonávat. Poté má administrátor prostor pro důležitější úlohy, které mají větší přínos pro firmu. Například produkt od firmy Oracle s názvem Oracle Autonomous Database [47] je zástupcem této technologie.

2.2 Webové aplikace

2.2.1 Popis architektury webových aplikací

Webová aplikace je počítačový program který za použití webového prohlížeče a webových technologií umožňuje vykonávat operace přes Internet. Při vývoji aplikace se používá kombinace

kódu na straně serveru (PHP [2], ASP [28]) a kódu na straně klienta (HTML [8], JavaScript [11]). Kód na straně serveru obstarává složitější operace, práci s úložištěm a klientský kód má za úkol prezentovat informace uživateli. Architektura webové aplikace popisuje komunikaci mezi jednotlivými komponentami aplikace. Podle rozložení aplikační logiky mezi klientem a serverem můžeme určit typ architektury. V dnešní době existují tři hlavní typy:

1. Single-Page Application(SPA) - Celou aplikaci představuje jediná stránka (odvozen název architektury), která se stáhne ze serveru. Klient poté už nemusí přejít na žádnou jinou stránku, nýbrž obsah stránky se dynamicky upravuje pomocí JavaScriptu. Při potřebě načíst nové data se využívá technologie AJAX [10]. Takováto aplikace má velmi dobré odezvy na klientské požadavky.

2. Microservice - Jsou malé, jednoduché služby, které mají za úkol vykonávat jednu funkcionalitu. Každá taková služba má poté vlastní HTTP [26] API. Toto API poté využívají vývojáři ve webové aplikaci. Tento typ architektury umožňuje urychlit vývoj a nasazení aplikací. Jednotlivé služby nejsou na sobě přímo závislé, proto také nemusí být nutně napsané pomocí stejného programovacího jazyku.

3. Serverless architektura - Tento pojem neznamená, že aplikace funguje naprosto bez serveru. Aplikace stálé pro vykonání kódu využívá server, ale úlohy související se správou a zřizování infrastruktury nejsou pro vývojáře viditelné. Tyto operace vykonává poskytovatel cloudových služeb. Díky tomuto přístupu se vývojáři mohou více zaměřit na obchodní logiku a poskytovat tak kvalitnější výsledky v tom co je pro filmu důležité.

2.2.2 Fungování webových aplikací

Většina webových aplikací je napsána za pomocí jazyka který je podporován webovými prohlížeči jako HTML [8] a JS [11]. Tyto jazyky využívají prohlížeč k vygenerování spustitelného programu. Některé aplikace jsou dynamické, potřebují zpracování informací na straně serveru. Ostatní zase můžou být statické bez jakékoliv potřeby zpracovávat informace na serveru. Ke své funkčnosti webové aplikace potřebují webový server který zpracovává požadavky od uživatele, aplikační server vykonávající požadované úlohy a většinou nějaký typ úložiště pro ukládání dat, nejčastěji se jedná o databázi.

Typický průběh webové aplikace:

1. Uživatel vytvoří požadavek na webový server přes internet za pomocí webového prohlížeče nebo uživatelského aplikačního rozhraní.

2. Webový server předá požadavek příslušnému aplikačnímu serveru.

3. Aplikační server vykoná potřebné operace - zpracování informací, dotazování na data a vytvoří výsledek pro požadované data.

4. Aplikační server zašle výsledek webovému serveru.

5. Webový server vrátí uživateli požadované data.

2.2.3 Výhody a nevýhody webové aplikace

Důležité je také zmínit, jak je na tom webová aplikace oproti desktopové. Webové aplikace mají několik výhod oproti desktopovým aplikacím. Protože webové aplikace běží uvnitř webového prohlížeče, vývojáři nemusí vyvíjet verze aplikace pro různé operační systémy. Například aplikace která běží v Chromu [20] bude fungovat jak na Windows [21] tak na macOS [22]. Další výhodou je že odpadá distribuování nové verze aplikace uživatelům. Stačí aktualizovat verzi na serveru a všichni uživatelé mají přístup k nejnovější verzi. Z uživatelského hlediska se dá za plus webové aplikace také považovat konzistentnost uživatelského rozhraní napříč více platformami, protože vzhled aplikace je závislý na webovém prohlížeči a ne na operačním systému. Za zmínku také stojí, že data zpracovávané aplikací se ukládají na vzdáleny server, tudíž uživatel má přístup k datům z různých zařízení, oproti přenášení souborů mezi počítači.

Samozřejmě webová aplikace nemá jen výhody, ale má také svoje nevýhody oproti desktopové aplikaci. Kvůli tomu že webové aplikace běží na sdíleném serveru, všichni uživatelé pracují se sdílenými prostředky serveru. Díky tomu může nastat situace, kdy při velkém vytížení serveru běží webová aplikace pomaleji. Proto aplikace na úpravu videa a fotografií mají většinou lepší výkon jako desktopové aplikace. To, že aplikace běží ve webovém prohlížeči jsem již zmínil jako výhodu, ale bohužel to platí i obráceně. Aplikace jsou zcela závislé na webovém prohlížeči. Z čehož vyplývá, pokud uživatel nemá přístup k internetu, nebo kvalita připojení je příliš nízká, tak není možné aplikaci spustit, případně chování aplikace bude příliš pomalé. Když v prohlížeči nastane chyba, tak například můžete přijít o neuložené data, nebo při aktualizaci prohlížeče mohou nastat neočekávané chyby.

3 Informační systém pro analýzu dat poruch v elektroenergetických sítích RelNet

Od roku 2000 se Vysoká škola báňská - Technická univerzita Ostrava stará o zpracování dat od některých distributorů elektrické energie v ČR a SR, které se používají pro výpočet spolehlivosti parametrů prvků sítě. V současné době se pro výpočet spolehlivosti používá informační systém RelNet [24] ve verzi webové aplikace. Aplikace slouží k výpočtu spolehlivostních parametrů prvků elektrické sítě a dotazovaní nad daty. Díky těmto výpočtů se mohou distributoři zaměřit na problematické části sítě a zlepšit tak své služby. Výpočty se provádějí nad záznamy, které jsou uložené v databázi spravovanou Fakultou elektrotechniky a informatiky. Přesnost výpočtů se zvyšuje s počtem záznamů v databázi.

3.1 Heterogenní data

Data potřebné k výpočtu spolehlivosti se získávají od jednotlivých distributorů elektrické energie. Bohužel v takovém to případě nastává problém že data jsou heterogenní. I přes stejný datový model, tato data nemusí být nutně shodné. Heterogenní data zapříčiňují složitou analýzu a zpracování. V důsledku tohoto problému bude použito jednotné relační schéma, které bude sloužit pro uložení dat. Nově vzniklé relace umožní analýzu a zpracování dat. Pro uložení dat o výpadcích byla vytvořena relace o 32 atributech. Tyto atributy byly pečlivě vybrány expertem v dané oblasti. Některé atributy fungují jako cizí klíče. Tyto atributy používají číselníkové hodnoty. Každý číselník obsahuje dvojici atributy Id a Value, kde hodnota Id je uložena v relaci pro výpadky [25].

3.2 Rámec pro uložení heterogenních dat

Hlavním cílem toho rámce je převod heterogenních dat do jednotného relačního schématu. Jakmile jsou data převedena, je možné se na ně dotazovat pomocí např. SQL [34].
Pro transformaci dat jsou nutné čtyři vstupní parametry.
Vstupní relace - data dodané distributorem, většinou soubor typu xls.
Transformační program - program pro transformaci vstupní relace na jednotnou relaci.
Číselníky - číselník je seznam dvojic (id,value). Číselníky jsou většinou vytvořeny Energetickým regulačním úřadem v případě České Republiky.
Překladové tabulky - sloužící k namapování různých hodnot distributora na jednotné hodnoty číselníků.

Transformace je vykonána pomocí transformačního programu, který je napsán jako XML [7] dokument. Pro každý atribut jednotné relace je vytvořen element `Transformation`. Seznam těchto elementů se nazývá transformační program. Tento program je vytvořen pro každý soubor dat od distributoru.

Průběh transformačního programu:

1. Přečtení jedno řádku ze vstupní relace.
2. Vyhodnocení elementu `Transformation` pro každý atribut jednotné relace.
3. Uložení výstupního záznamu.
4. Jestli vstupní relace neobsahuje řádky program se ukončí, v opačném případě se pokračuje od kroku jedna.

Ukázkový element Transformation (viz výpis 1) slouží k transformaci dvou hodnot vstupních atributů na jeden výstupní atribut. Konkrétně hodnota 5. atributu vstupního záznamu udává čas a hodnota na 4. udává datum. Tyto dvě hodnoty se poté převedou na hodnotu výstupního atributu, která reprezentuje datum a čas začátku události. Na obrázku 1 je příklad vstupních dat. Pro první řádek bude transformace provedena takto: Hodnotu atributu `cas_por` převedeme na hodiny a minuty - 8:23. Z atributu `dat_por` se hodnota převede na rok, měsíc a den - 23.3.2001. Poté se tyto hodnoty spojí na jeden výstupní atribut - 23.3.2001 8:23.

```
<Transformation OutputItemIndex="18">
      <Date>
        <FormattedDate FormattingString="HH':'mm">
          <InputItem TupleName="a" Index="5"/>
        </FormattedDate>
        <FormattedDate FormattingString="yyyyMMdd">
          <InputItem TupleName="a" Index="4"/>
        </FormattedDate>
      </Date>
</Transformation>
```

Výpis 1: Ukázka transformace pro jeden atribut

ev_cislo	ev_rok	vn	dat_por	cas_por	dat_kman	cas_kman	dat_kon	cas_kon	lj	ned_prace	e1_e2	pricina	zarizeni	p_zariz
51	2001	5571	20010423	08:23			20010423	16:31	0	0	E2	211	121	042
88	2001	519	20010529	16:34	20010529	16:36	20010605	20:18	126	3,644	E2	211		
131	2001	556	20010717	01:03			20010719	13:52	0	0	E2	341	72	183
163	2001	523	20010802	09:34			20010802	15:10	0	0	E2	211	221	051
172	2001	519	20010804	17:33	20010804	17:35	20010914	13:40	92,3	3,076	E2	342	121	013
172	2001	519	20010804	17:33	20010804	17:35	20010914	13:40	92,3	3,076	E2	342	121	051
182	2001	5574	20010804	23:32	20010804	23:38	20010805	18:00	100	10,04	E2	341	221	046
250	2001	1318	20010913	08:48	20010913	08:53	20010913	17:45	282	19,975	E2	217	221	021

Obrázek 1: Ukázka kolekce vstupních dat

Výpis 2 představuje transformaci hodnoty 12. atributu vstupního záznamu pomocí číselníku 13.

```
<Transformation OutputItemIndex="31">
  <CodeBookValue CodeBookName="13">
      <InputItem TupleName="a" Index="12"/>
  </CodeBookValue>
</Transformation>
```

Výpis 2: Ukázka transformace pro číselníkový atribut

3.3 Datová analýza aplikace

Původní verze aplikace používala databázový systém RadegastDB. Aktuální verze aplikace používá dva druhy databázového systému. Pro uložení uživatelských informaci se používá databázový systém SQLite [23], který obsahuje relace `User` (viz tabulka 1), `Role` (viz tabulka 2), `StoredSetting` (viz tabulka 3), `FormType` (viz tabulka 4), `AssignedRole` (viz tabulka 5). Samotná data pro výpočty jsou potom uložena v databázovém systému SQL Server.

Popis tabulek

Tabulka `User` obsahuje přihlašovací údaje uživatele, jméno, příjmení a datum posledního přihlášení. Tabulka `Role` slouží jako číselník pro dostupné role v systému. V tabulce `StoredSetting` je uloženo nastavení uživatele pro jednotlivé formuláře systému. Tabulka `FormType` slouží jako číselník formulářů aplikace. V tabulce `AssignedRole` se uchovávají informace o rolích, kterými uživatel disponuje.

Lineární zápis těchto tabulek je následující [1]

User (Login, Password, FName, LName, LastLogin)
Role (IdRole, Name)
StoredSetting (*IdFormType*, *Login*, SerializedValue)
FormType (IdFormType, TypeName)
AssignedRole (*Login*, *IdRole*)

Datový slovník

Atribut	Dat.Typ	Velikost	Klíč	Null	Popis
Login	TEXT		PK	N	Přihlašovací jméno
Password	TEXT		N	N	Heslo
FName	TEXT		N	A	Jméno
LName	TEXT		N	A	Příjmení
LastLogin	TEXT		N	A	Datum posledního přihlášení

Tabulka 1: Schéma tabulky User

Atribut	Dat.Typ	Velikost	Klíč	Null	Popis
IdRole	Integer	8	PK	N	Číslo role
Name	TEXT		N	N	Název role

Tabulka 2: Schéma tabulky Role

[1] Legenda: **tabulka**, primární klíč, *cizí klíč*

Atribut	Dat.Typ	Velikost	Klíč	Null	Popis
IdFormType	Integer	8	PK,FK	N	Číslo typu formuláře
Login	TEXT		PK,FK	N	Přihlašovací jméno
SerializedValu	TEXT		N	N	Nastavení

Tabulka 3: Schéma tabulky StoredSetting

Atribut	Dat.Typ	Velikost	Klíč	Null	Popis
IdFormType	Integer	8	PK	N	Číslo formuláře formuláře
TypeName	TEXT		N	N	Název formuláře

Tabulka 4: Schéma tabulky FormType

Atribut	Dat.Typ	Velikost	Klíč	Null	Popis
Login	TEXT		PK,FK	N	Uživatelské jméno
IdRole	Integer	8	PK,FK	N	Číslo role

Tabulka 5: Schéma tabulky AssignedRole

Druhý typ databázového systému, který aplikace používá, je MS SQL Server. Tento systém pracuje s databázi poruch prvků v elektroenergetických sítích. V databázi se nachází relace číselníků (viz tabulka 6), tabulka `Outage` (viz tabulka 9) a `Passportization` (viz tabulka 8).

Relace c_01 až c_13 slouží jako číselníky pro cizí klíče v relaci `Outage` (viz tabulka 9). Všechny relace číselníků mají totožné schéma. V tabulce 6 je uveden popis všech číselníků. Dále jako příklad uvádím jen číselník `c_01` (viz tabulka 7). Relace `Outage` obsahuje data poruchy prvku, nad kterými se provádí výpočty. V tabulce `Passportization` (viz tabulka 8) jsou uloženy informace o počtu prvků distributorů.

Číselník Id	Název
c_01	REAS
c_02	Typ události
c_03	Rozvodna
c_04	Napětí
c_05	Druh sítě
c_06	Příčina události
c_07	Druh zařízení
c_08	Poškozené zařízení
c_09	Druh zkratu
c_10	Typ poškozeného zařízení
c_11	Výrobce
c_12	Souhrnné údaje
c_13	Druh poruchy

Tabulka 6: Jednotlivé číselníky databáze

Lineární zápis těchto tabulek je následující [2]

c_01 (Id, Value)
Passportization (d_p_s, equipment, equipment_voltage, year, reas, value)
Outage (*distributor*, event_order, *event_type*, *distribution_point*, area, *network_type*, *network_voltage*, *equipment_voltage*, original_event_order, *event_cause*, *equipment_type*, *damaged_ equipment*, *damaged_equipment_type*, amount, *short_type*, *producer*, production_date, T0, T1, T2, T3, T4, TZ, P1, P2, D1, D2, Z1, Z2, LxT, failure_type)

Datový slovník

Atribut	**Dat.Typ**	**Velikost**	**Klíč**	**Null**	**Popis**
Id	Integer		PK	N	Číslo hodnoty
value	NChar	150	N	A	Hodnota číselníku

Tabulka 7: Schéma tabulky c_01

Atribut	**Dat.Typ**	**Velikost**	**Klíč**	**Null**	**Popis**
d_p_s	Char	1	PK	N	Hodnota udávající číselník
equipment	Integer		PK	N	Kód z číselníku c_07 nebo c_08
equipment_voltage	Integer		PK	N	Napětí
year	Integer		PK	N	Rok
reas	Integer		PK	N	Číslo distributora
value	Float		N	A	Množství zařízení

Tabulka 8: Schéma tabulky Passportization

[2] Legenda: **tabulka**, primární klíč, *cizí klíč*

Atribut	Dat.Typ	Velikost	Klíč	Null	Popis
distributor	Integer		FK	N	Číslo distributora
event_order	BigInteger		N	A	Číslo události
event_type	Integer		FK	N	Typ události
distribution_point	Integer		FK	A	Rozvodna
area	Integer		N	A	Napájecí oblast
network_type	Integer		FK	A	Druh sítě
network_voltage	Integer		FK	N	Napětí sítě
equipment_voltage	Integer		FK	N	Napětí zařízení
original_event_order	Integer		N	A	Číslo původní události
event_cause	Integer		FK	A	Příčina události
equipment_type	Integer		FK	A	Druh zařízení
damaged_equipment	Integer		FK	A	Poškozené zařízení
damaged_equipment_type	Integer		FK	A	Typ poškozeného zařízení
amount	Integer		N	A	Množství
short_type	Integer		FK	A	Druh zkratu
producer	Integer		FK	A	Výrobce
production_date	Integer		N	A	Rok výroby
T0	Datetime2	7	N	N	Datum a čas začátku události
T1	Datetime2	7	N	A	Datum a čas začátku manipulací
T2	Datetime2	7	N	A	Datum a čas konce manipulací pro vymezení poruchy
T3	Datetime2	7	N	A	Datum a čas obnovení dodávky v úseku ovlivněném událostí
T4	Datetime2	7	N	A	Datum a čas konce události
TZ	Datetime2	7	N	A	Datum a čas zemního spojení
P1	Integer		N	A	Výkon v čase T0 v kVA
P2	Integer		N	A	Výkon v čase T2 v kVA
D1	Integer		N	A	Počet DTS bez napětí v čase T0
D2	Integer		N	A	Počet DTS bez napětí v čase T2
Z1	Integer		N	A	Počet zákazníků bez napětí v čase T0
Z2	Integer		N	A	Počet zákazníků bez napětí v čase T2
LxT	Integer		N	A	Výkon kráte čas
failure_type	Integer		FK	N	Druh poruchy

Tabulka 9: Jednotné schéma Outage

3.4 Popis funkcí formulářů

Následující odstavce popisují funkce jednotlivých formulářů.
Pasportizace - formulář slouží k výpočtu pasportizačních hodnot dle souboru, jehož obsahem bude podle zvolených atributů uživatelem (zařízení, distribuční oblast, rok) seznam zařízení, které má zvolený distributore ve své sítu a k počtu zařízení přiřadí počty poruch ve vybraném roce, vypočítá intenzitu a střední dobu trvání.

Prvková spolehlivost - slouží k získaní dokumentu ve kterém je k vybranému prvku, napětí, zvolenému období a distribuční společnosti je zobrazená tabulka, ve které je podle roku a společnosti zobrazeny počty poruch, pasportizace, délka období, součin, intenzita poruch, trvání všech poruch a střední doba trvání všech poruch. Soubor také obsahuje grafy intenzity poruch a střední dobu trvání poruch pro jednotlivé společnosti, roky a celkové období.

Spolehlivostní parametry - má podobnou funkci jako Prvková spolehlivost s rozdílem, že u toho exportovaného dokumentu je vypočítána jen intenzita poruch a střední doba trvání poruch pro zvolený prvek. Do toho výpočtu se berou v potaz všechny společnosti a veškeré sledované období.

Příčina události - vyexportuje soubor s tabulkou a grafy kde je podle vybraného období a společnosti znázorněn počet příčin událostí.

Četnosti trvání poruch - vygeneruje soubor, který obsahuje trvání poruch rozdělené do pěti intervalů a u každého intervalu je číslo reprezentující počet poruch v daném časovém rozmezí. Tato informace je také znázorněna grafem.

Přednastavení výpočtů - Tento formulář slouží k nastavení minimální a maximální hodnoty doby trvání událostí.

Dotazování - umožňuje zobrazit, exportovat události a jejich trvání podle vybraných kritérií. Ve formuláři Nastavení dotazů je možné nastavovat hodnoty atributů.

4 Analýza a implementace

Hlavním úkolem této práce bylo navrhnout a implementovat vytváření jednotné databáze s uživatelským rozhraním do stávající webové aplikace.

V současné době je možné vytvoření databáze pouze pomocí desktopové aplikace, což podstatně ztěžuje práci administrátorů a přidává zbytečné kroky na víc do již stávajícího procesu. Pokud je zapotřebí aktualizovat databázi, administrátor je nucen nejprve indexovat data pomocí desktopové aplikace. Tato data jsou následně uložena v databázovém systému RadegastDB, který je vyvíjen výzkumnou skupinou Katedry informatiky. Data se poté musí naimportovat do databáze webové aplikace.

Integrace vytváření jednotné databáze do webové aplikace výrazně zjednoduší proces aktualizace databáze. V případě potřeby se administrátor bude moct jednoduše přihlásit do aplikace, nahrát potřebné soubory a databázi aktualizovat. Důležitým požadavkem pro systém je možnost indexace vybraných souboru, které obsahují data poruch. S tímto požadavkem také úzce souvisí návrh vhodného uživatelského, které bude umožňovat správu jednotlivých souborů.

4.1 Datová analýza

Součástí mé práce byl návrh relace OutageTempTable (viz tabulka 10), která slouží pro uložení nepotvrzených zpracovaných záznamů. Tato relace má schéma totožné jako relace Outage (viz tabulka 9) s tím rozdílem, že obsahuje navíc tři nové atributy FileName, UserLogin a ErrorText. Schéma relace Outage (viz tabulka 9) se rozšířilo o atribut FileName, který je totožný s atributem FileName v relaci OutageTempTable (viz tabulka 10). Další relace, která byla navržena, se nazývá IndexFiles. Ta slouží k uložení informací o souborech (viz tabulka 11).

Poslední navržená relace je ErrorSummaryTable (viz tabulka 12). V této relaci se bude uchovávat přehled chybně indexovaných záznamů. Na SQL Server [4] budou také přemístěny tabulky uchovávající informace o uživatelích - User (viz tabulka 1), Role (viz tabulka 2), StoredSetting (viz tabulka 3), FormType (viz tabulka 4), AssignedRole (viz tabulka 5).

Lineární zápis těchto tabulek je následující [3]

Legenda: **tabulka**, primární klíč, *cizí klíč*
IndexFiles (FileName, Index, records, indexRecords, IndexDate)
ErrorSummaryTable (UserLogin, FileName, ErrorCount,Accepted)

Datový slovník

[3] Legenda: **tabulka**, primární klíč, *cizí klíč*

Atribut	Dat.Typ	Velikost	Klíč	Null	Popis
FileName	Varchar	70	N	N	Název souboru
UserLogin	varchar	50	N	N	Uživatelské jméno
ErrorText	varchar	500	N	N	Popis chyby

Tabulka 10: Schéma tabulky OutageTempTable

Atribut	Dat.Typ	Velikost	Klíč	Null	Popis
FileName	Varchar	70	PK	N	Název souboru
Index	Bit		N	N	Hodnota zda byl soubor indexován
Records	Integer		N	A	Počet záznamu v souboru
IndexRecords	Integer		N	A	Počet indexovaných záznamu
IndexDate	Datetime		N	A	Datum indexace

Tabulka 11: Schéma tabulky IndexFiles

Atribut	Dat.Typ	Velikost	Klíč	Null	Popis
UserLogin	Varchar	100	PK	N	Uživatelské jméno
FileName	Varchar	100	N	N	Název souboru
ErrorCount	Integer		N	N	Počet chyb
Accepted	Bit		N	N	Zda byl soubor schválen

Tabulka 12: Schéma tabulky ErrorSummaryTable

4.2 Funkční analýza

Seznam funkcí systému:

1. Nahrání adresáře dat
2. Indexace dat
3. Nahrání číselníků
4. Vymazání databáze

Pro každou funkci byl sestav scénář případu užití.

4.2.1 Scénáře případu užití

Název: Nahrání adresáře dat
Prekondice:

Postkondice:
Hlavní scénář:

1. Uživatel vybere adresář pro nahrání, potvrdí nahrání
2. Systém zkontroluje soubory a nahraje soubory

Alternativní scénář

2. Soubory na serveru se liší od nahrávaných
   (a) Systém zobrazí uživateli přehled souborů a možnosti nahrání
   (b) Uživatel zvolí jednu možnost
   (c) Systém provede vybranou akci

Název: Indexace dat
Prekondice: Přihlášený administrátor
Postkondice:
Hlavní scénář:

1. Uživatel vybere adresář nebo soubor pro indexaci
2. Systém zkontroluje existenci dat,provede indexaci a zobrazí data
3. Uživatel vybere požadovanou akci
4. Systém provede vybranou akci

Alternativní scénář

2. V databázi se nacházejí nepotvrzené data
   (a) Systém zobrazí uživateli jestli chce data potvrdit nebo zahodit
   (b) Uživatel vybere požadovanou akci
   (c) Systém provede vybranou akci

Název: Nahrání číselníků
Prekondice:
Postkondice:
Hlavní scénář:

1. Uživatel vybere soubory pro nahrání
2. Systém načte soubory a aktualizuje číselníky

Alternativní scénář
Název: Vymazání databáze
Prekondice: Přihlášený administrátor
Postkondice:
Hlavní scénář:

1. Uživatel vybere možnost vymazání databáze
2. Systém vymaže databázi

Alternativní scénář

4.2.2 Detailní popis funkcí

Nahrání adresáře dat

Uživatel má možnost nahrávat adresář dat pomocí prvku formuláře, který slouží k nahrávání celých adresářů, nebo může nahrávat jednotlivé soubory v přehledu souborů. Před nahráním souborů se provede kontrola, zda na serveru již neexistuje soubor se stejným jménem. V případě, že je takovýto soubor nalezen, zkontroluje se obsah obou souborů. Jestliže se obsah souborů neshoduje, je přidán do kolekce odlišných souborů. Po kontrole souborů se uživateli zobrazí formulář, kde jsou vypsány jednotlivé soubory. Na tomto formuláři má uživatel možnost zvolit jestli chce přepsat všechny soubory, přepsat jen starší soubory, nebo může ručně vybrat soubory které se mají přepsat. V případě, že uživatel nevybere žádnou z nabízených možností, soubory se nenahrají.

Indexace dat

Indexaci dat může uživatel uskutečnit buď pro konkrétní soubor, adresář nebo všechny adresáře na serveru. Před samotnou indexací dat se provede kontrola, jestli v databázi nejsou pro požadovaný soubor nepotvrzené data. Pokud se v databázi taková data nachází systém zobrazí uživateli formulář s možnosti tyto data zpracovat. Poté co se data zpracují uživatel znova vybere indexaci. Po úspěšné indexaci se uživateli zobrazí tabulka s korektně zpracovanými záznamy, druhá tabulka obsahující nekorektně zpracovaná záznamy a přehled chyb pro jednotlivé soubory. Takto zpracovaná záznamy jsou uloženy v nově vytvořené tabulce. Záznamy poté uživatel může potvrdit, v takovém to případě se data vloží do hlavní tabulky v databázi nebo se data dají zahodit, čímž se vymažou z databáze.

Nahrání číselníků

Uživatel nejprve vybere soubory číselníku. Poté potvrdí svůj výběr stiskem tlačítka. Jednotlivé soubory se přečtou a aktualizují se číselníky v databázi. V případě že číselník neexistuje je vytvořen.

Vymazání databáze

Funkce slouží k vymazání všech tabulek v databázi, kromě číselníku.

4.3 Návrh a implementace

4.3.1 Komponenty systému

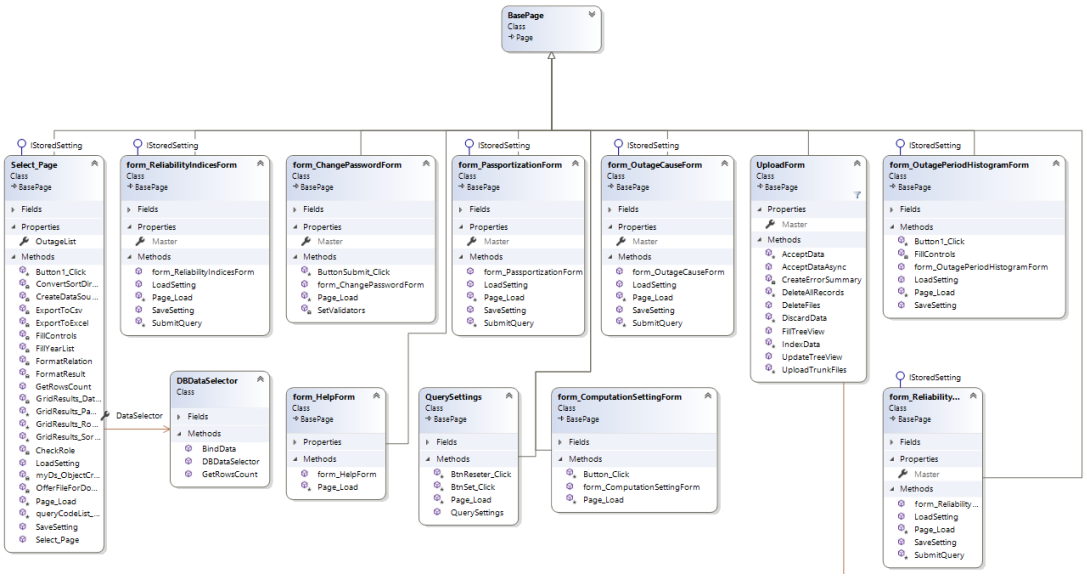
Aplikace se nově skládá ze tří částí:

1. Prezentační vrstva - v této části jsou zobrazeny získané data z logické a databázové vrstvy.
2. Logická vrstva - nově implementovaná část systému, která se stará o správu souboru a indexaci dat.
3. Databázová vrstva - umožňuje přístup do databáze. Tato část je využívaná aplikační a logickou vrstvou.

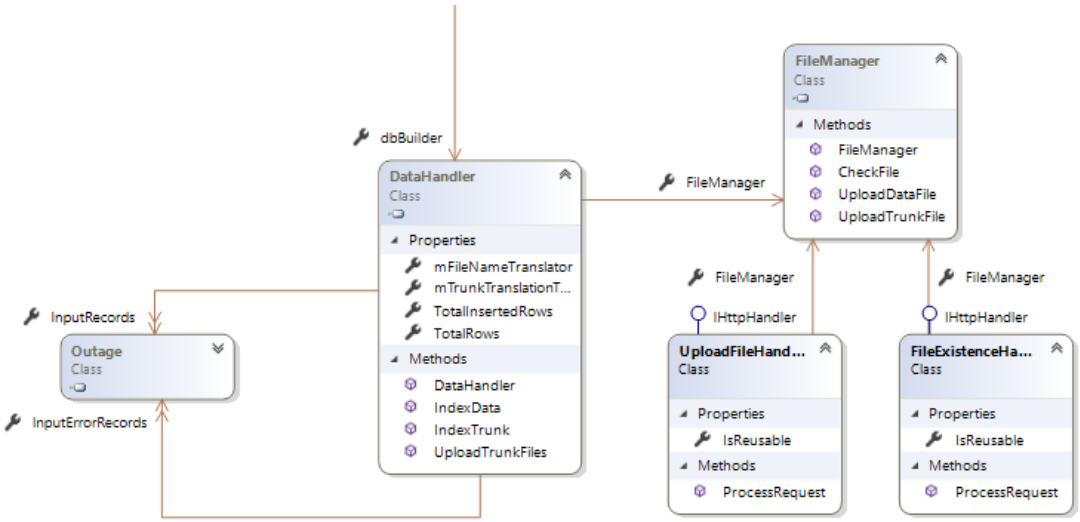
Zde jsem do systému přidal dříve neimplementovanou logickou vrstvu a snažil jsem se tak přiblížit třívrstvé architektuře. Prezentační vrstva má však stále přístup do databázové vrsty a díky tomu nesplňuje všechny kritéria této architektury.

4.3.2 Třídní diagram logické a prezentační vrstvy

Pro přehlednost byl třídní diagram rozdělen na dvě části. Obrázek 2 zobrazuje strukturu formulářů v aplikaci. Třídní diagram na obrázku 3 ukazuje statickou strukturu systému části, která se stará o vytvoření databáze. Třída `UploadForm` reprezentuje formulář. Tato třída se vytváří při každém HTTP [26] požadavku od uživatele. `DataHandler` je třída starající se o zpracovaní přijatých dat (indexace dat). Tato třída využívá pro zpracování dat třídy převzaté z desktopové aplikace, u kterých bylo nutné implementovat úpravy pro webovou aplikaci. `FileManager` slouží k vykonání procesů, které souvisí se správou souborů. Dále byly vytvořeny třídy `FileExistenceHandler` a `UploadFileHandler`. Obě třídy slouží k zpracování asynchronních požadavků od klienta. `FileExistenceHandler` slouží k ověření nahrávaných souborů a `UploadFileHandler` se stará o požadavek nahrání souborů na server.



Obrázek 2: Třídní digram prezentační vrstvy



Obrázek 3: Třídní digram části systému pro vytvoření databáze

K implementaci byly vyžity programy SQL Server Management Studio [18] a Microsoft Visual Studio [29]. SSMS bylo použito při tvorbě nově navržených relací. Pro úpravu kódu bylo použito Visual Studio.

4.3.3 Datová vrstva

V aplikaci je již naimplementované ORM fungující na návrhovém vzoru Data mapper. Tudíž nebyl důvod datovou vrstvu nijak měnit, jenom byly přidány potřebné CRUD operace pro nově vytvořené relace a další potřebné SQL [34] dotazy. Například funkce 3 sloužící pro nahrávání dat číselníku a příkazy na obrázku 4, které vloží nově indexované data do databáze. Níže popsané úlohy se provádějí v transakci, pro zajištění integrity dat.

Funkce 3 je volána uvnitř logické vrstvy, je součástí případu užití nahrání číselníků (viz 4.2.2) a spouští se pro každý nahraný soubor reprezentující číselník. Nejprve se data ze souboru nahrají do dočasné tabulky a poté se zkontroluje jestli v databázi existuje číselník. Pokud číselník neexistuje tak se vytvoří a vloží se do něj všechna data. V případě existence číselníku se aktualizují stávající hodnoty a vloží se nové.

```
CREATE TABLE #c_01(
Id int NOT NULL,
value varchar(200) NULL
)
insert into #c_01 values (1,'REAS1')
insert into #c_01 values (2,'REAS2')
insert into #c_01 values (3,'REAS3')
insert into #c_01 values (4,'REAS4')
insert into #c_01 values (5,'REAS5')
If OBJECT_ID('c_01') is null
   begin
      create table c_01 (
      id int not null primary key,
      [value] varchar(200) not null)

      insert into c_01
      select * from #c_01

   end
else
   begin
      update c_01
      set c_01.value = #c_01.value
      from c_01 join #c_01 on c_01.Id = #c_01.Id

      insert into c_01
      select #c_01.* from #c_01 left join c_01 on c_01.Id = #c_01.Id
      where c_01.Id is null
   end
```

Výpis 3: Funkce pro vložení číselníků

```
        delete
        from Outage
        where Outage.file_name in ('tl_reas02_2001_01-12_3.xml','tl_reas02_2002_01-12_2.xml')

        INSERT INTO Outage
                SELECT distributor
                   ,event_order
                   ,event_type
                   ,distribution_point
                   ,area
                   ,network_type
                   ,network_voltage
                   ,equipment_voltage
                   ,original_event_order
                   ,event_cause
                   ,equipment_type
                   ,damaged_equipment
                   ,damaged_equipment_type
                   ,amount
                   ,short_type
                   ,producer
                   ,production_date
                   ,T0
                   ,T1
                   ,T2
                   ,T3
                   ,T4
                   ,TZ
                   ,P1
                   ,P2
                   ,D1
                   ,D2
                   ,Z1
                   ,Z2
                   ,LxT
                   ,failure_type
                   ,file_name
                   FROM OutageTempTable
                   where [file_name] in ('tl_reas02_2001_01-12_3.xml','tl_reas02_2002_01-12_2.xml') and errorText is null

delete from ErrorSummaryTable
  where Accepted = 1 and [File] in ('tl_reas02_2001_01-12_3.xml','tl_reas02_2002_01-12_2.xml')

  update IndexFiles
  set   indexRecords = (select count(*) from OutageTempTable where [file_name] = [FileName] and errorText is null), [index] = 1,
        IndexDate = GETDATE(),records = isnull((select ErrorCount from ErrorSummaryTable where [File] = [FileName]),0) +
        isnull((select count(*) from OutageTempTable where [file_name] = [FileName] and errorText is null),0)
  where [fileName] in ('tl_reas02_2001_01-12_3.xml','tl_reas02_2002_01-12_2.xml')

  update ErrorSummaryTable
  set Accepted = 1
  where [File] in ('tl_reas02_2001_01-12_3.xml','tl_reas02_2002_01-12_2.xml')

  delete from OutageTempTable
 where [file_name] in ('tl_reas02_2001_01-12_3.xml','tl_reas02_2002_01-12_2.xml')
```

Výpis 4: Příkazy použité pro vložení indexovaných dat

SQL příkazy z výpisu 4 jsou součástí případu užití Indexace dat (viz 4.2.2), spustí se v momentě kdy uživatel potvrdí indexované data. Nejprve se odstraní staré záznamy z relace Outage (viz tabulka 9). Jako další krok se provede vložení nových záznamů a vymažou se záznamy o chybách. Následuje aktualizace relace IndexFiles (viz tabulka 11) a ErrorSummaryTable (viz tabulka 12). Posledním krokem je smazání záznamů z relace OutageTempTable (viz tabulka 10).

4.3.4 Prezentační vrstva

Uživatelské rozhraní bylo vytvořeno pomocí webového formuláře, který nabízí technologie ASP.Net WebForms [19]. Tato technologie umožňuje prvky formuláře přidávat dvěma způsoby. Buď se dají prvky formuláře napsat přímo do šablony, nebo pomocí návrháře se mohou přetáhnout z nabídky elementů. Jednotlivé prvky formuláře se při vykreslení převedou přímo na HTML [8] elementy. Obrázek 4 je ukázka zdrojového kódu formuláře, který obsahuje prvky technologie ASP.Net.

```
        <div style="margin-top: 10px; display: flex;">
            <asp:Label runat="server" ID="indexResult" Visible="False" Style="margin-left: 20px;" ViewStateMode="Enabled"></asp:Label>
            <asp:Button Class="form-control" runat="server" ID="acceptButton" Text="Potvrdit změny" OnClick="AcceptData" Style="font-weight: bold; color: #ffffff; font-family: Verdana, Aria
            <asp:Button Class="form-control" runat="server" ID="deleteTmpbtn" Text="Zahodit změny" OnClick="DiscardData" Style="font-weight: bold; color: #ffffff; font-family: Verdana, Aria
        </div>
        <asp:Button Class="form-control" runat="server" ID="deleteBtn" Text="Vymazat všechny záznamy v databázi" OnClick="DeleteAllRecords" Style="margin-top: 10px; display: inline; font-we
        <div style="margin-top: 10px;">
            <asp:FileUpload runat="server" ID="uploadDir" Style="display: inline; padding-left: 20px;" webkitdirectory multiple />
            <asp:Button Class="btn" runat="server" ID="uploadDirButton" Text="Nahrát adresář dat" Style="width: 200px; font-weight: bold; color: #ffffff; font-family: Verdana, Arial; backgr
        </div>
        <div style="margin-top: 10px;">
            <asp:FileUpload runat="server" ID="uploadDataFile" AllowMultiple="true" Style="display: inline; padding-left: 20px;" />
            <asp:Button Class="btn" runat="server" ID="uploadDataFileButton" Text="Nahrát číselníky" OnClick="UploadTrunkFiles" Style="width: 200px; font-weight: bold; color: #ffffff; font-
        </div>
        <br>
        <asp:Button Class="form-control" runat="server" ID="buttonAll" Text="Indexace všech adresářů" OnClick="IndexData" data-type="" data-path="" Style="margin-top: 10px; display: inline;
        <br>
    </div>

    <div style="flex-grow: 1;">
        <p>Statistiky</p>
        <asp:Label runat="server" ID="lblRecordCount" Style="margin-left: 20px;">Počet záznamů v databázi: </asp:Label>
        <br>
        <asp:Label runat="server" ID="lblErrorRecordCount" Style="margin-left: 20px;">Počet nekorektně extrahovaných záznamů: </asp:Label>
        <br>
        <asp:Label runat="server" ID="lblLastSaveDate" Style="margin-left: 20px;">Datum posledního uložení souboru: </asp:Label>
        <br>
        <asp:Label runat="server" ID="lblLastIndexDate" Style="margin-left: 20px;">Datum poslední indexace dat: </asp:Label>
        <br>
        <asp:Label runat="server" ID="lblTlFile" Style="margin-left: 20px;">Počet uložených tl souborů:<asp:Label runat="server" ID="lblFileCount" Style="margin-left: 20px;"> </asp:Label>
        </asp:Label>
    </div>
</div>
```

Obrázek 4: Zdrojový kód formuláře

Na této vrstvě jsou také implementovány funkce pro asynchronní volaní metod. Na obrázku 5 je ukázka funkce napsané v JS, která používá rozhraní fetch pro zavolání `FileExistenceHandler`.

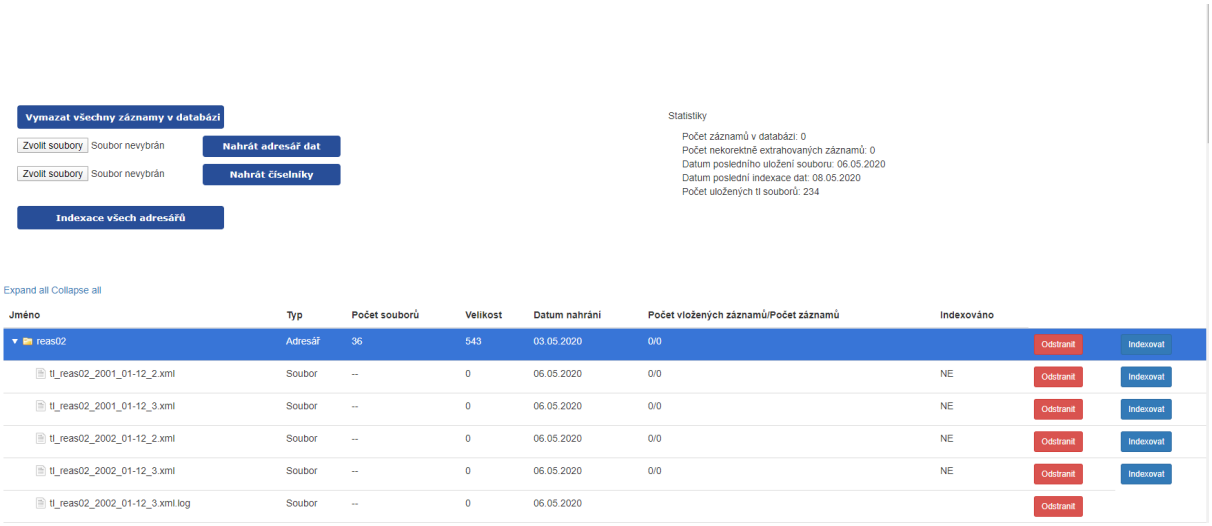
```
let responseCheck = await fetch('/FileExistenceHandler.ashx', {
    method: 'POST',
    body: formData,
})
let resultCheck = await responseCheck.json()
```

Obrázek 5: Ukázka funkce pro vytvoření požadavku

4.4 Popis formulářů

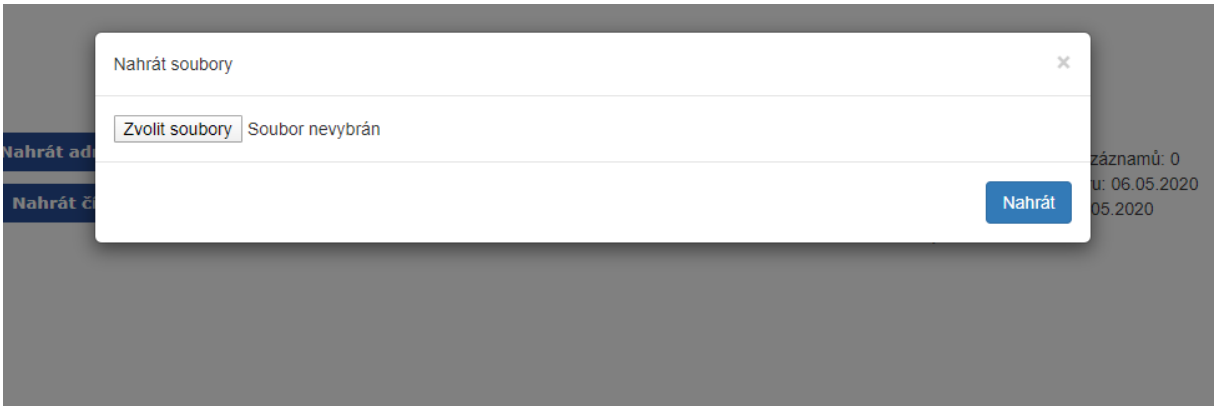
Na obrázku číslo 6 je formulář pro nahrávání a indexaci dat. Formulář obsahuje tlačítka pro indexaci souborů, adresářů, nahrávání adresářů se soubory, nahrávání souborů pro číselníky. V pravé části jsou zobrazeny statistiky databáze. Spodní část slouží k zobrazení obsahu adresáře kde se ukládají soubory pro vytvoření databáze. Výpis souboru je zobrazen po jednotlivých složkách s možnosti danou složku rozkliknout a zobrazit obsah adresáře. U adresářů se ve výpisu zobrazují také informace o celkovém počtu souborů, velikosti, datum nahrání, počet vložených

záznamů v databázi, tlačítko na odstranění adresáře a tlačítko pro indexaci. Soubory ve výpisu navíc obsahují informaci jestli je daný soubor zaindexovaný.



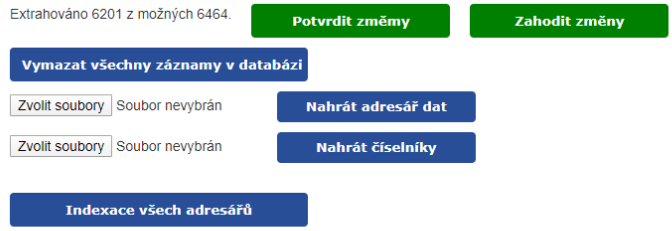
Obrázek 6: Formulář pro indexaci dat

Při kliknutí pravým tlačítkem na výpis souborů se uživateli nabídne možnost nahrát soubory a zobrazí se formulář na obrázku 7. Umístění nahraného soubory je odvozeno podle toho v jakém adresáři uživatel kliknul.



Obrázek 7: Dialogové okno pro nahrávání

Obrázek 8 a 9 ukazují formulář po indexaci dat. Zobrazí se tlačítka pro potvrzení a zahození změn. Dále se na stránce zobrazí tři tabulky. První tabulka obsahuje pro jednotlivé soubory počet chyb. Zbylé dvě tabulky jsou pro zobrazení extrahovaných záznamů.



Soubor	Chyba	Počet
tl_reas02_2001_01-12_2.xml	Chyba ciselniku 7.	2
tl_reas02_2002_01-12_2.xml	Chybí hodnota EventType.	91
	Chyba ciselniku 6.	1
tl_reas02_2003_01-12_2.xml	Chyba ciselniku 8.	2
tl_reas02_2004_01-11_3.xml	Chybí hodnota EventType.	166
	Chybí hodnota FailureType.	1

Obrázek 8: Formulář po indexaci dat

Chybně extrahované záznamy (263):

Pozice	REAS	Číslo události	Typ události	Rozvodna	Napájecí oblast	Druh sítě	Napětí sítě	Napětí zařízení	Číslo původní události	Příčina události	Druh zařízení	Poškozené zařízení	Typ poškozeného zařízení	Množství	Druh zkratu	Výrobce	Rok výroby	T
754	2	814	1	-E-	350	-E-	5	5	-E-	24	5710	8	-E-	-U-	-E-	-E-	-DE-	05.07.20
1282	2	1374	1	-E-	-E-	-E-	5	5	-E-	24	5710	8	-E-	-U-	-E-	-E-	-DE-	15.09.20
178	2	358	1	-E-	-E-	-E-	5	5	-E-	3141	57	8	-E-	-U-	-E-	-E-	-DE-	21.03.20
1055	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	11.12.20
1056	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	14.12.20
1057	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	15.12.20
1058	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	16.12.20
1059	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	16.12.20
1060	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	17.12.20
1061	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	17.12.20
1062	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	17.12.20
1063	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	17.12.20
1064	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	18.12.20
1065	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	18.12.20
1066	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	18.12.20
1067	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	18.12.20
1068	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	18.12.20
1069	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	18.12.20
1070	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	19.12.20
1071	2	-E-	-E-	-E-	-E-	-E-	5	5	-E-	-E-	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	19.12.20

12345678910...

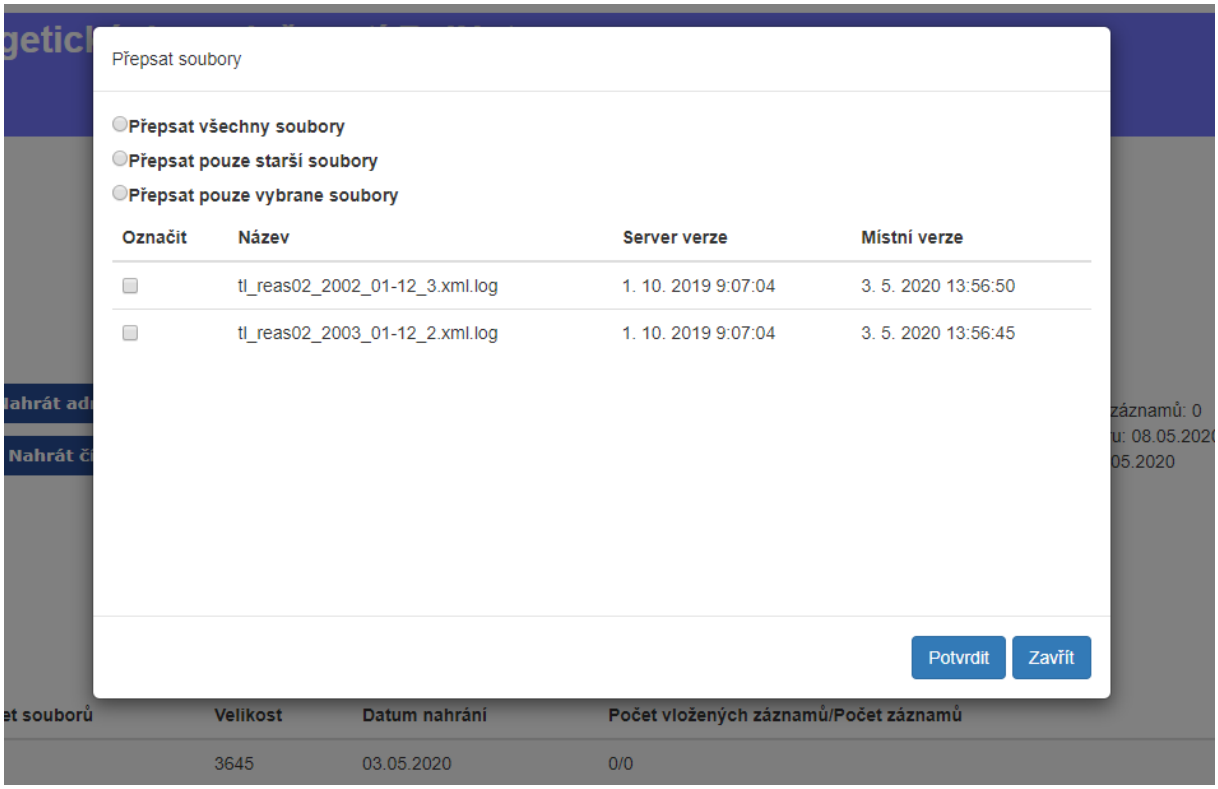
Korektně extrahované záznamy (6201):

Pozice	REAS	Číslo události	Typ události	Rozvodna	Napájecí oblast	Druh sítě	Napětí sítě	Napětí zařízení	Číslo původní události	Příčina události	Druh zařízení	Poškozené zařízení	Typ poškozeného zařízení	Množství	Druh zkratu	Výrobce	Rok výroby	
1	2	1	1	-E-	159	-E-	5	5	-E-	21	1	2	-E-	-U-	-E-	-E-	-DE-	(
2	2	2	1	-E-	192	-E-	5	5	-E-	29	-E-	-E-	-E-	-U-	-E-	-E-	-DE-	(
3	2	3	1	-E-	126	-E-	5	5	-E-	9	2	5	-E-	-U-	-E-	-E-	-DE-	(
4	2	4	1	-E-	367	-E-	5	5	-E-	9	2	5	-E-	-U-	-E-	-E-	-DE-	(
5	2	5	1	-E-	387	-E-	5	5	-E-	21	1	101	-E-	-U-	-E-	-E-	-DE-	(
6	2	6	1	-E-	351	-E-	5	5	-E-	9	57	8	-E-	-U-	-E-	-E-	-DE-	(
7	2	7	1	-E-	839	-E-	5	5	-E-	21	1	101	-E-	-U-	-E-	-E-	-DE-	(
8	2	8	1	-E-	17	-E-	5	5	-E-	22	1	5	-E-	-U-	-E-	-E-	-DE-	(
9	2	9	1	-E-	188	-E-	5	5	-E-	21	1	101	-E-	-U-	-E-	-E-	-DE-	(
10	2	10	1	-E-	189	-E-	5	5	-E-	29	57	15	-E-	-U-	-E-	-E-	-DE-	(
11	2	11	1	-E-	33	-E-	5	5	-E-	21	1	5	-E-	-U-	-E-	-E-	-DE-	(
12	2	12	1	-E-	35	-E-	5	5	-E-	32	1	-E-	-E-	-U-	-E-	-E-	-DE-	(
13	2	13	1	-E-	155	-E-	5	5	-E-	21	1	5	-E-	-U-	-E-	-E-	-DE-	(
14	2	14	1	-E-	37	-E-	5	5	-E-	21	3	6	-E-	-U-	-E-	-E-	-DE-	(
15	2	15	1	-E-	838	-E-	5	5	-E-	29	57	-E-	-E-	-U-	-E-	-E-	-DE-	(
16	2	16	1	-E-	188	-E-	5	5	-E-	33	1	2	-E-	-U-	-E-	-E-	-DE-	(
17	2	17	1	-E-	184	-E-	5	5	-E-	29	1	15	-E-	-U-	-E-	-E-	-DE-	(

Obrázek 9: Formulář po indexaci dat

Pokud při nahrávání souboru existuje na serveru verze, která není totožná s nahrávanou,

zobrazí se uživateli formulář pro volbu souboru 10. Uživatel si může vybrat jednu ze tří možností nahrání. Pokud si uživatel nevybere žádnou z nabízených možností soubory se nenahrají.



Obrázek 10: Formulář pro vybraní souboru

4.5 Popis použitých technologií

4.5.1 Microsoft SQL Server

SQL Server [4] je relační systém řízení báze dat vyvíjen firmou Microsoft [42]. Podobně jako ostatní relační SŘBD je postaven na základě SQL, standardním programovacím jazykem pro komunikaci s relační databází. Vývojářů nabízí také procedurální nástavbu jazyka SQL, Transact-SQL (T-SQL) [17]. T-SQL je Microsofti implementace jazyka SQL, která přidává sadů různých programových konstrukcí.Pro správu SQL Serveru Microsoft nabízí integrované prostředí SQL Server Management Studio (SSMS)[18]. SSMS nabízí řadu nástrojů pro konfiguraci, řízení a administraci SQL Serveru.

4.5.2 Microsoft Visual Studio

Microsoft Visual Studio [29] je integrované vývojářské prostředí od firmy Microsoft [42]. Používá se pro psaní desktopových aplikací, webových stránek, webových aplikací a webových služeb. Obsahuje editor kódu, debugger, nástroje na tvorbu grafického uživatelského rozhraní, editor na tvorbu databázového schématu a také podporuje nejrozšířenější verzovací systémy. Dostupné je

buď v bezplatné verzi Community, která je určena pro jednotlivce a velmi malé firmy, nebo také existují placené verze Professional a Enterprise pro větší podniky.

4.5.3 .Net Framework

.Net Framework [13] je platforma používaná pro vývoj softwaru od firmy Microsoft [42]. Tato platforma byla vytvořena za účelem vytváření aplikaci které poběží na operačním systému Windows [21]. První verze .Net Frameworku byla vydána v roce 2002. První verze nesla označení .Net framework 1.0. Od této verze prošel .Net framework řadou úprava a vylepšení a v dnešní době je k dispozici verze .NET Framework 4.8. Může být používán jak pro vývoj desktopových aplikací tak pro vývoj webových aplikací. Jedna z výhod tohoto frameworku je podpora různých programovacích jazyků např. C# [30], Visual Basic [31], F# [32]. Díky tomuto si může vývojář zvolit jazyk ve kterém vyvíjet.

Architektura .Net Framework

.Net Framewokr se skládá ze dvou důležitých komponent:

1. Common Language Runtime (CLR) je komponenta která se stará o běžící aplikace. Poskytuje různé druhy služeb jako je správa vláken, automatická správa paměti, prací s výjimkami a další.

2. .NET Framework Class Library nabízí rozhraní pro čtení a zápis souboru, komunikaci s databází, kreslení a další. Dále také obsahuje datové typy pro řetězce, čísla, data a mnoho dalších.

Kód aplikace napsané v jednom z možných jazyků je zkompilován do Common Intermediate Language (CIL). Takto zkompilovaný kód je poté uložen v souboru s příponou .dll nebo .exe (assembly).
Při spuštění aplikace komponenta CLR vezme assembly a použije just-in-time compiler (JIT) k převodu CIL na strojový kód, který může být spuštěn na daném počítači.

4.5.4 Hypertext Markup Language

Hypertext Markup Language, neboli take HTML [8] je značkovací jazyk používaný pro tvorbu webových stránek. HTML není programovací jazyk tudíž neumožňuje tvořit dynamické prvky, ale slouží k organizaci a formátování dokumentu. Každý HTML dokument se skládá z tagů (elementů), které slouží jako stavební kámen daného dokumentu. Tyto tagy tvoří strukturu dokumentu.

4.5.5 Cascading Style Sheets

Cascading Style Sheets zkráceně CSS [9] je jazyk sloužící k specifikaci jaká bude výsledná grafická podoba dokumentu. Dokument je většinou textový soubor napsaný s využitím HTML [8], ale může být také použit jiný jazyk jako SVG [15] nebo XML [7]. Pomocí CSS můžeme ovlivňovat rozložení a vzhled dokumentu. Patří mezi jazyky založené na pravidlech. Nadefinujeme pravidlo které se má použit pro určitý element nebo skupinu elementů.

4.5.6 JavaScript

JavaScript [11] je programovací jazyk, který se většinou používá na straně klienta a umožňuje implementaci složitějších funkcionalit na webové stránce. Dodává stránce dynamičnost. Umožňuje dynamicky upravovat strukturu HTML [8] dokumentu za pomocí Document Object Model(DOM) [16]. DOM je programátorské rozhraní pro HTML [8] a XML [7]. Slouží k reprezentaci stránky takovým způsobem, aby programy mohly upravovat strukturu, vzhled a obsah dokumentu. Dále umožňuje reagovat na uživatelské akce (kliknutí myší, zmáčknutí klávesy...), posílat požadavky přes Internet, nastavovat cookies atd. S kombinací CSS [9] a HTML je to základní nástroj pro tvorbu webových stránek.

4.5.7 AJAX

AJAX [10] je akronym pro Asynchronní JavaScript [11] a XML [7]. Popisuje techniky které se používají při tvorbě webových aplikací. Hlavní přínosem těchto technik je možnost aktualizace webové stránky asynchronně, což znamená že uživatelský prohlížeč nemusí načíst celou stránku znovu, když je zapotřebí aktualizovat jen malou část. X v názvu znamená XML. Značkovací jazyk pomocí kterého se ukládají informace při komunikaci. V dnešní době se však místo XML také velmi hojně používá JSON [33]. JSON (JavaScript Object Notation) je textový formát založený na jazyce JavaScript, který se používá pro přenos dat. Jakákoliv datové struktura se převede do řetězce a následně je tento řetězec použit pro komunikaci. Na rozdíl od XML neobsahuje názvy jednotlivých elementů, ale je použita kolekce párů název/hodnota, případně seznam hodnot. Zápis je velice dobře čitelný pro člověka a také pro stroje k analýze a generování.

4.5.8 Bootstrap

Bootstrap [12] je jeden z nejpopulárnějších frameworku pro tvorbu webových stránek. Kombinuje v sobě HTML [8],CSS [9] a JavaScript [11]. Mezi nesporné výhody které tento framework přináší je zajištění kompatibility napříč různými prohlížeči. Dále také umožňuje vytvářet stránky, které se automatický přizpůsobí různým typů zařízení určených k prohlížení webu. Díky obrovské komunitě kterou tento framework má se neustále vytváří nové funkcionality pro zlepšení uživatelského požitku.

4.5.9 BootboxJS

BootboxJS [14] je malá JavaScriptova [11] knihovna umožňující vytvářet dialogové okno za pomocí Bootstrap [12] frameworku, bez toho aniž by se vývojář musel zajímat o vytvoření, odstranění a správu DOM [16] elementů nebo JavaScriptove události.

5 Závěr

V rámci této práce jsem shrnul obecné informace týkající se databázových systémů. Přiblížil jsem také relační databázové systémy, které se v dnešní době řadí mezi nejpoužívanější typ databázových systémů. Dále jsem se v první kapitole popsal architekturu webových aplikací, jejich základní principy a fungování. Současně se zde objevují informace o nejpoužívanějších typech těchto aplikací.

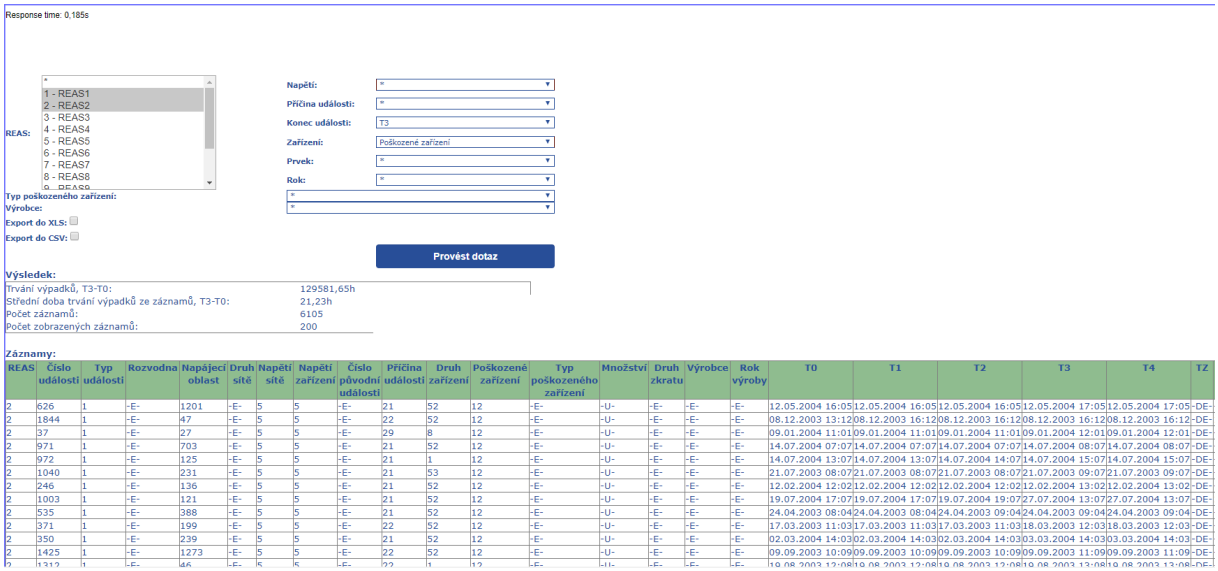
Následně jsem nastudoval fungování informačního systém pro analýzu dat poruch v elektrických sítích a databázi poruch, který spravuje Katedra informatiky Vysoké školy báňské - Technická univerzita Ostrava a shrnul jeho funkcionalitu. Zde jsem také uvedl datovou analýzu systému.

Za cíl své práce jsem měl úkol implementování vytváření databáze ve webové aplikaci. Této části jsem se věnoval v poslední kapitole. Na tomto místě jsem ve webové aplikaci, která umožňovala výpočty spolehlivostních parametrů prvků, nově přidal část systému, která implementuje toto vytváření databáze.

Pro zpracování se stále používá stávající algoritmus z desktopové aplikace, jenž byl upraven pro potřeby webové aplikace. Dále jsem vytvořil uživatelské rozhraní pro správu souborů a indexaci dat. Nyní má uživatel možnost nahrát a indexovat soubory přímo ve webové aplikaci.

Veškerou implementaci jsem v poslední části zdokumentoval. Provedl jsem datovou analýzu a vytvořil jsem datový slovník pro nově vytvořené tabulky v databázi. Jsou zde také zaznamenány důležité případy užití týkající se této nové funkcionality webové aplikace.

Na závěr jsem provedl testování nahráním souborů a indexací dat na stroji ASUS VivoBook S (Intel Core i7-8550U, 16 GB RAM) s lokálním SQL Serverem. Nahrál jsem celkem 9 adresářů s celkovým počtem souborů 1351, které obsahují data o poruchách, pasportizační data, číselníky a transformační dokumenty. Největší adresář s velikosti 265 MB byl nahrán za 9 sekund. Zpracováno bylo 488 953 záznamů za přibližně 37 minut, z toho bylo 331 401 korektních a 157 552 nekorektních záznamů. Na obrázku 11 můžeme vidět, že po zaindexování nových dat je dotazování možné.



Obrázek 11: Formulář pro dotazování

Zdrojové kódy aplikace se především nacházejí na serveru dbsys.cs.vsb.cz ve složce `C:\inetpub\wwwroot\Relnet3`.

6 Literatura

1. MySQL. MySQL [online]. c2020, Oracle Corporation [cit. 10.05.2020]. Dostupné z: https://www.mysql.com/

2. PHP: Hypertext Preprocessor. PHP: Hypertext Preprocessor [online]. c2001 [cit. 10.05.2020]. Dostupné z: https://www.php.net/

3. Databázový software a aplikace | Microsoft Access. Databázový software a aplikace | Microsoft Access [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-365/access

4. SQL Server 2019 | Microsoft. SQL Server 2019 | Microsoft [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://www.microsoft.com/cs-cz/sql-server/sql-server-2019

5. FileMaker. Inovace na pracovišti – FileMaker, dceřiná společnost společnosti Apple [online]. c2019 [cit. 2020-05-10]. Dostupné z: https://www.filemaker.com/cz/

6. DBase. DBase [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://www.dbase.com/

7. XML. XML Core Working Group Public Page [online]. c1997-2007 [cit. 2020-05-10]. Dostupné z: https://www.w3.org/XML/

8. HTML. Web Hypertext Application Technology Working Group (WHATWG) [online]. c2018 [cit. 2020-05-10]. Dostupné z: https://html.spec.whatwg.org/multipage/

9. CSS. World Wide Web Consortium (W3C) [online]. 2020 [cit. 2020-05-10]. Dostupné z: https://www.w3.org/Style/CSS/specs.en.html

10. AJAX. MDN Web Docs [online]. 2020 [cit. 2020-05-10]. Dostupné z: https://developer.mozilla.n-US/docs/Web/Guide/AJAX/Getting_Started

11. JavaScript. Standard ECMA-262 [online]. c1999 [cit. 2020-05-10]. Dostupné z: https://www.ecma-international.org/publications/standards/Ecma-262.htm

12. Bootstrap. Bootstrap · The most popular HTML, CSS, and JS library in the world. [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://getbootstrap.com/

13. .NET Framework. Oficiální domovská stránka Microsoft [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://dotnet.microsoft.com/

14. BootboxJS. Bootbox.js — alert, confirm and flexible dialogs for the Bootstrap framework [online]. c2011-2019 [cit. 2020-05-10]. Dostupné z: http://bootboxjs.com/

15. SVG. World Wide Web Consortium (W3C) [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://www.w3.org/TR/SVG2/

16. Document Object Model. World Wide Web Consortium (W3C) [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://www.w3.org/DOM/DOMTR

17. Transact-SQL. Transact-SQL Reference (Database Engine) [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver15

18. SQL Server Management Studio. Oficiální domovská stránka Microsoft [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15

19. ASP.NET Web Forms. Oficiální domovská stránka Microsoft [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://dotnet.microsoft.com/apps/aspnet/web-forms

20. Google Chrome. Webový prohlížeč Google Chrome [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://www.google.com/intl/cs_CZ/chrome/

21. Windows 10 – Microsoft Store Česká republika. Oficiální domovská stránka Microsoft [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://www.microsoft.com/cs-cz/store/b/windows

22. Catalina. Apple (Česká republika) [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://www.apple.com/cz/macos/catalina/

23. SQLite. SQLite Home Page [online]. c2019 [cit. 2020-05-10]. Dostupné z: https://www.sqlite.org/index.html

24. GOŇO, Radomír, Michal KRÁTKÝ a Stanislav RUSEK. Analysis of Distribution Network Failure Databases. Przeglad elektrotechniczny. Warszawa: SIGMA-NOT, 2010,86(8), s. 168- 171. ISSN 033-2097

25. Kratky, M., Gono, R., Rusek, S., Dvorsky, J. A framework for an analysis of failures data in electrical power networks, Proc. PEA Conf. on Power, Energy, and Applications, Gaborone, BW, (2006), p. 45-46.

26. HTTP. World Wide Web Consortium (W3C) [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://www.w3.org/Protocols/Specs.html

27. Oracle Database. Oracle Česká republika | Integrated Cloud Applications and Platform Services [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://www.oracle.com/cz/database/

28. Active Server Pages. Oficiální domovská stránka Microsoft [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms526064(v%3Dvs.90)

29. Visual Studio. Oficiální domovská stránka Microsoft [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://visualstudio.microsoft.com/cs/

30. Dokumentace k jazyku C#. Oficiální domovská stránka Microsoft [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/csharp/

31. Dokumentace k Visual Basic. Oficiální domovská stránka Microsoft [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/visual-basic/

32. Dokumentace k F#. Oficiální domovská stránka Microsoft [online]. c2020 [cit. 2020-05-10]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/fsharp/

33. Introducing JSON. JSON [online]. c2017 [cit. 2020-05-10]. Dostupné z: https://www.json.org/json-cz.html

34. MOLINARO, Anthony. SQL Cookbook. Sebastopol: O'Reilly Media, 2005. ISBN 0596009763.

35. Co je autonomní databáze? Oracle Česká republika | Integrated Cloud Applications and Platform Services [online]. c2020 [cit. 2020-05-11]. Dostupné z: https://www.oracle.com/cz/database/what-is-autonomous-database.html

36. Most Widely Deployed and Used Database Engine. SQLite Home Page [online]. c2019 [cit. 2020-05-12]. Dostupné z: https://www.sqlite.org/mostdeployed.html

37. ANSI. ANSI latest headlines [online]. c2020 [cit. 2020-05-12]. Dostupné z: https://www.ansi.org/

38. Android. Android | The platform pushing what's possible [online]. c2020 [cit. 2020-05-12]. Dostupné z: https://www.android.com/

39. IOS 13. Apple (Česká republika) [online]. c2020 [cit. 2020-05-12]. Dostupné z: https://www.apple.com/cz/ios/ios-13/

40. KRATKÝ, Michal a Radim BAČA. Databázové systémy. 2009 [cit. 2020-05-12]. VŠB – Technická univerzita Ostrava.

41. GARCIA-MOLINA, Hector, Jeffrey D. ULLMAN a Jennifer WIDOM. Database Systems: The Complete Book. 2nd Edition. New Jersey: Prentice Hall Press, 2008. ISBN 9780131873254.

42. Microsoft. Oficiální domovská stránka Microsoft [online]. c2020 [cit. 2020-05-13]. Dostupné z: https://www.microsoft.com/cs-cz/

43. CODD E. F., Relational Completeness of Data Base Sublanguages, Prentice-Hall, 1970.

44. Databáze. Co je to databáze | Oracle Česká Republika [online]. c2020 [cit. 2020-05-13]. Dostupné z: https://www.oracle.com/cz/database/what-is-database.html#WhatIsDBMS

45. Berkeley, University of California. Home | University of California, Berkeley [online]. c2020 [cit. 2020-05-14]. Dostupné z: https://www.berkeley.edu/

46. IBM. IBM - Czech Republic [online]. [cit. 2020-05-14]. Dostupné z: https://www.ibm.com/cz-en?lnk=m

47. Co je autonomní databáze? Co je to autonomní databáze | Oracle Česká Republika [online]. c2020 [cit. 2020-05-15]. Dostupné z: https://www.oracle.com/cz/database/what-is-autonomous-database.html

A Seznam příloh

- $\backslash ui \backslash ecsafety - webapp$ - složka obsahující projekt pro prezentační a logickou vrstvu
- $\backslash ui \backslash ecsafety - common$ - složka obsahující projekt datové vrstvy
- *cpcomponent.dll* - knihovna
- *wnvxls.dll* - knihovna
- *wnvxls.xml* - konfigurační soubor